



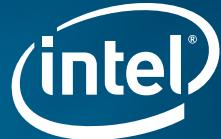
インテル® Parallel Advisor を使用 したシリアル・アプリケーション のモデリング



スレッディング・アシスタントであるインテル® Parallel Advisor の hotspot 解析により、Microsoft* Visual Studio* C++ の既存のソースコードで並列化すべき箇所をより簡単に見つれます。

革新的なこのツールは、並列化の調査から、並列化によりメリットが得られるコード領域の特定、並列化候補のパフォーマンスと正当性の評価までのプロセスをガイドします。

この重要な洞察を基に、優れた設計判断を下し、開発サイクルの早い段階で問題を解決することが可能になります。



はじめに

本ガイドでは、インテル® Parallel Studio 2011 を使用して、シリアル・アプリケーションを並列化する方法を説明します。最初に、サンプルコードを使って実際にインテル® Parallel Advisor を操作しながら、その強力な機能について説明します。その後、ユーザー自身でほかのインテル® Parallel Studio 2011 のコンポーネントを試してみてください。最後のセクションには、スレッド化に役立つ重要な情報が含まれています。

インテル® Parallel Advisor

インテル® Parallel Advisor は、既存のアプリケーションの並列化を設計する際に役立つツールセットです。リソースを割り当てる前に、パフォーマンスの影響を評価し、コード領域の並列化により生じるコストをリファクタリングすることができます。並列化候補のコード領域をマークするだけなので、アプリケーションはシリアルのままであります。そのため、並列コードへの移行に伴う変更を既存のテストシステムで確認することができます。

実践

ステップ 1: インテル® Parallel Studio 2011 のインストール

1. インテル® Parallel Studio の評価版を[ダウンロード](#)します。
2. parallel_studio_2011_setup.exe をクリックして、インテル® Parallel Studio をインストールします。

カスタム・インストールを選択する場合は、必ずインテル® Parallel Advisor をインストールしてください。

ステップ 2: Tachyon_Advisor サンプル・アプリケーションの準備とロード

サンプル・アプリケーションの準備：

1. tachyon_Advisor.zip サンプルファイルの場所を確認します。デフォルトでは、C:\Program Files\Intel\Parallel Studio 2011\Advisor\samples\en にあります。このサンプルは、Microsoft® Visual Studio® 2005 を使用して作成された C++ コンソール・アプリケーションです。

インテル® Parallel Advisor のサンプルは、Microsoft® Visual Studio® 2005、2008、2010 で使用できます。

2. tachyon_Advisor.zip を書き込み可能なディレクトリー (C:\Work\Samples フォルダーなど) またはシステムの共有ディレクトリーに展開します。

インテル® Parallel Studio 2011 は、将来に渡って利用できるハイパフォーマンスな並列アプリケーションを開発するための製品スイートです。以下のコンポーネントで構成されています。

- › インテル® Parallel Advisor - 既存の C/C++ アプリケーションにおける並列化の機会の特定と評価を支援します。
- › インテル® Parallel Composer - Windows® ベースのクライアント・アプリケーションの並列化に取り組む開発者にインテル® C++ コンパイラ、ライブラリー、デバッグ機能を提供します。
- › インテル® Parallel Amplifier - プロセッサー・アーキテクチャやアセンブリー・コードの知識がなくても、マルチコア・パフォーマンスのボトルネックを簡単に素早く発見することができるパフォーマンス・アナライザー / チューニング・ソリューションです。
- › インテル® Parallel Inspector - エラーやソフトウェアの不具合を簡単に素早く見つけるシリアル / 並列コードの動的メモリー / スレッド解析ツールです。
- › インテル® スレッディング・ビルディング・ブロック (インテル® TBB) - 受賞歴もある C++ テンプレート・ライブラリーで、スレッドをタスクに抽象化し、信頼性と移植性に富んだスケーラブルな並列アプリケーションを作成できます。

シリアル・アプリケーションのモデリング



Microsoft* Visual Studio* で tachyon サンプルを開く：

1. [File (ファイル)] > [Open (開く)] > [Project/Solution… (プロジェクト / ソリューション…)] を選択して tachyon_Advisor.sln ファイルを参照し、Microsoft* Visual Studio* にロードします (図 1)。
2. tachyon_Advisor ソリューションには、いくつかのプロジェクトが含まれています。このガイドでは、1_tachyon_serial プロジェクトと tachyon.common プロジェクトのみを変更します。その他のプロジェクト (2_tachyon_annotated、3_tachyon_cilk、3_tachyon_tbb) は本ガイドでは使用しません (図 2)。
3. 1_tachyon_serial は、インテル® Parallel Advisor のインクルード・ディレクトリーを参照するよう設定されています。設定を確認するには、[Solution Explorer (ソリューション エクスプローラー)] でプロジェクトを右クリックして、[Properties (プロパティ)] を選択します。図 3 のハイライトされた箇所で設定されています。
4. Tachyon Advisor サンプルのコードを理解するために、図 4 をお読みください。

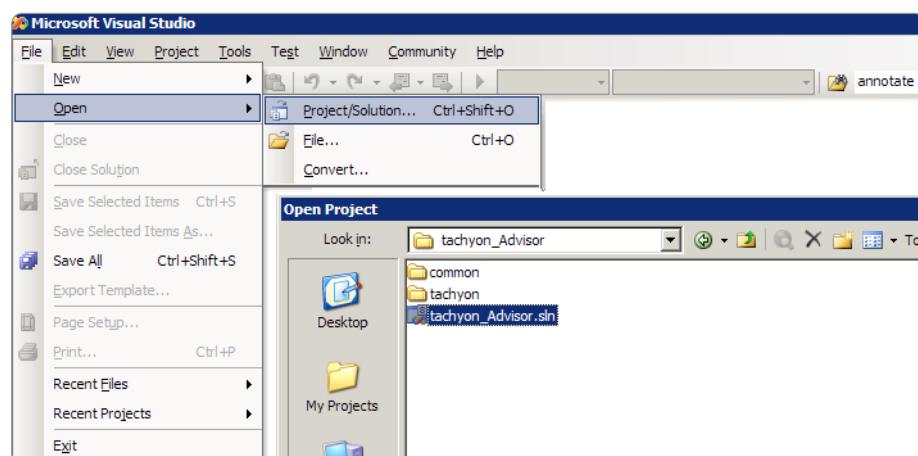


図 1

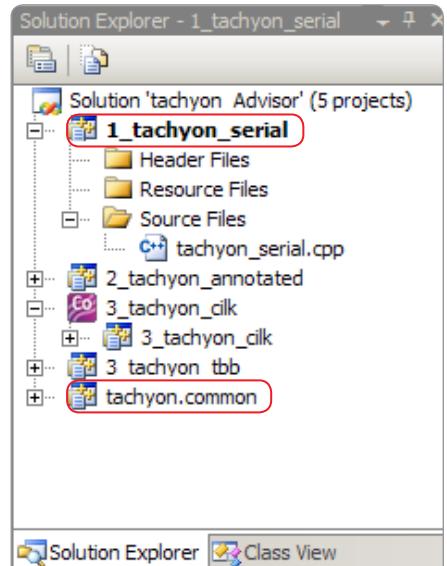


図 2

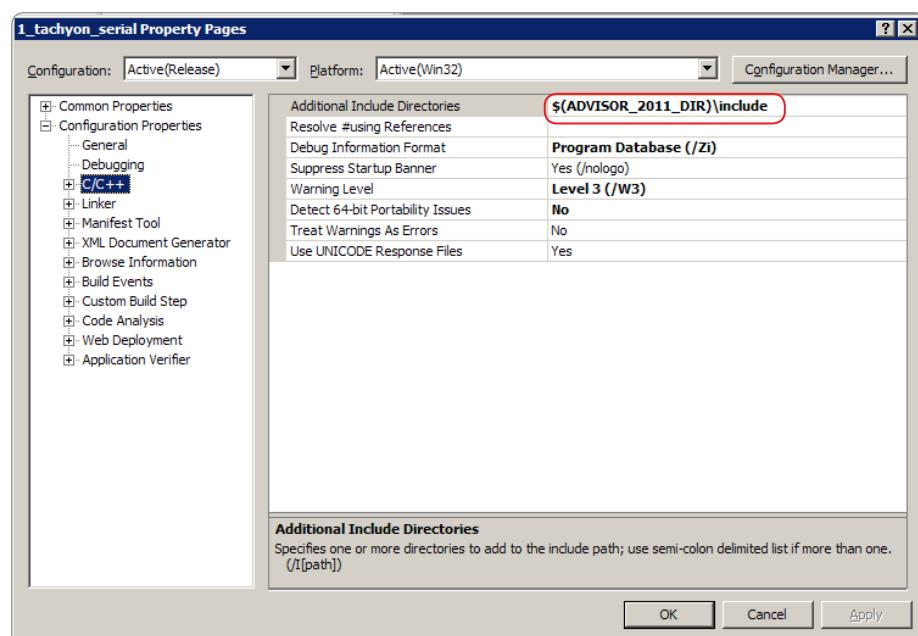


図 3

tachyon コードサンプルについて

Tachyon は、2-D レイトレーシング / レンダリング・プログラムです。イメージと経過時間 (約 10 秒) を計算して、表示します。その後、ウィンドウの内容を確認できるように 5 秒間待ってから、ウィンドウを閉じます。

ソースの tachyon_serial.cpp では、3 つの関数を使用しています。 thread_trace() は、parallel_thread() が図を描画する前にスタティック変数を初期化します。 render_one_pixel() は出力する色を決定します。 parallel_thread() と render_one_pixel() のループに注目してください。

本ドキュメントでは、インテル® Parallel Advisor を使用して、これらの関数を並列化する方法を説明します。

シリアル・アプリケーションのモデリング



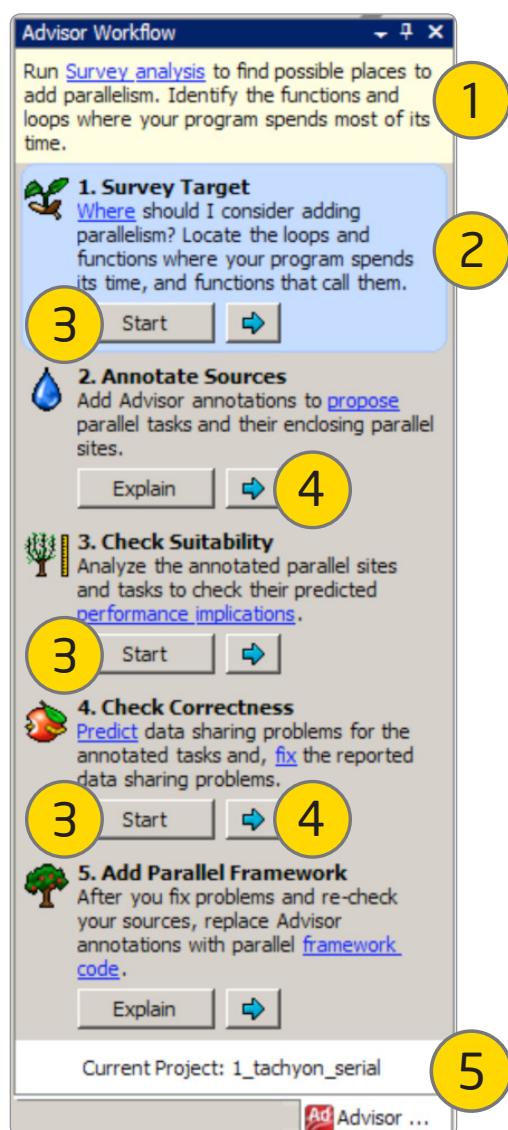
インテル® Parallel Advisor の起動

インテル® Parallel Advisor は、Microsoft® Visual Studio® のプラグインとして、ツールバー、Microsoft® Visual Studio® のツールメニュー、または [Solution Explorer (ソリューション エクスプローラ)] のプロジェクト・コンテキスト・メニューから起動できます。インテル® Parallel Advisor ツールバーでハイライトされてるアイコンをダブルクリックします (図 5)。

インテル® Parallel Advisor の [Advisor Workflow (Advisor ワークフロー)] ウィンドウが開きます。[Advisor Workflow (Advisor ワークフロー)] ウィンドウは、並列化の検証手法をガイドします (図 6)。



図 5



- 1 次の操作のヒント
- 2 現在のステップはハイライトされています。
- 3 インテル® Parallel Advisor ツールを起動するためのショートカット
- 4 インテル® Parallel Advisor ウィンドウを開き、ツールレポートを表示するためのショートカット
- 5 現在のスタートアップ・プロジェクト

図 6

シリアル・アプリケーションのモデリング



調査対象：並列化の利点が得られる呼び出し位置とループの特定

まず、インテル® Parallel Advisor の調査ツールを実行して、プログラム時間を最も消費している呼び出し位置とループを特定します。並列化の検証候補として時間がかかっているコールツリーとループに重点を置きます。

ソリューションの [Start-Up Project (スタートアップ プロジェクト)] として 1_tachyon_serial を定義します。

1. Release 構成でビルドします。
2. ツールバー[ボタン]からインテル® Parallel Advisor の [Advisor Workflow (Advisor ワークフロー)] ウィンドウを開きます。
3. [1. Survey Target (1. 調査対象)] にある [Start (開始)] ボタンをクリックします (図 7)。

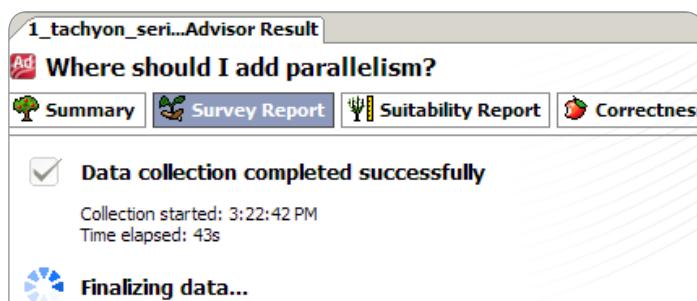
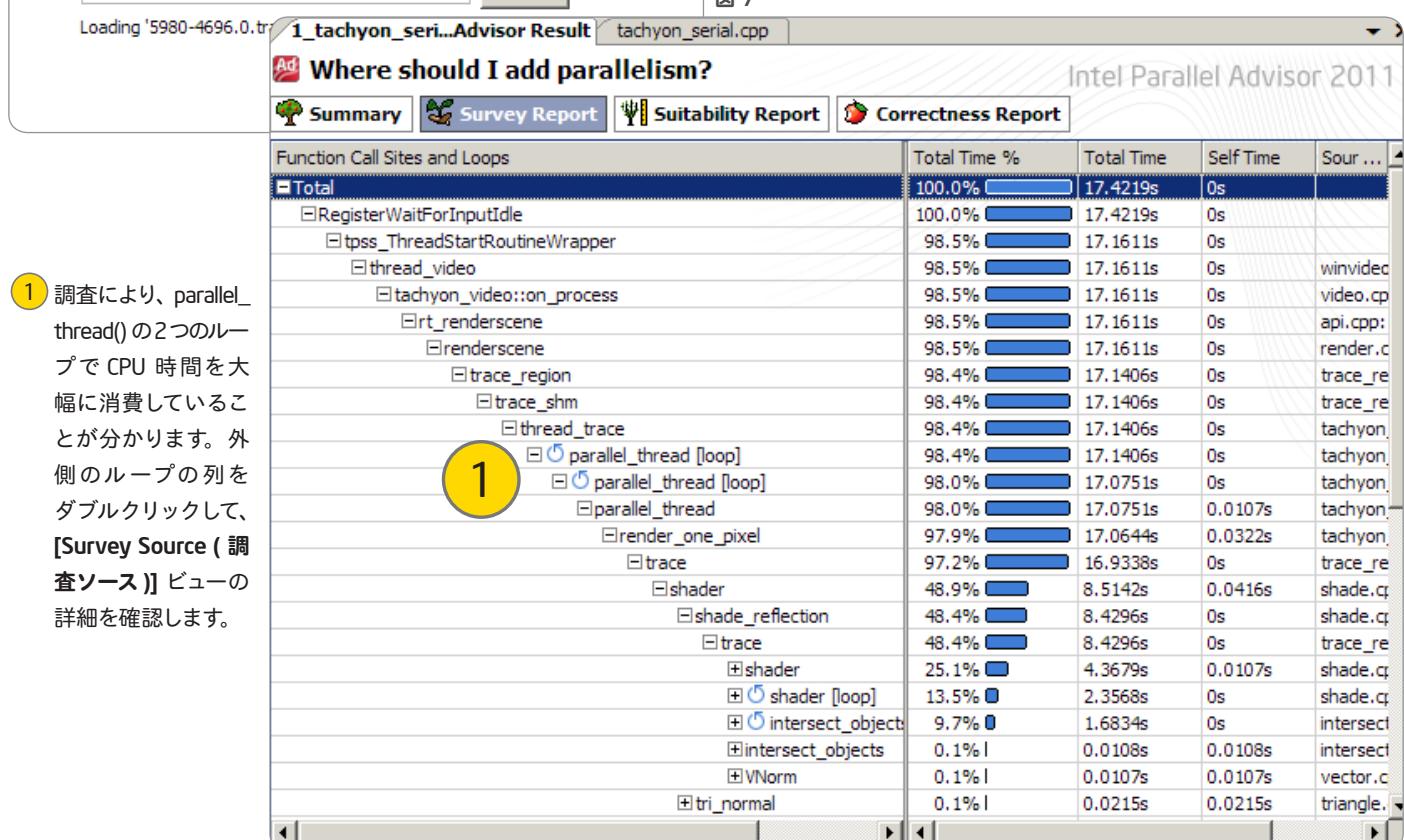


図 7



- 1 調査により、parallel_thread() の 2 つのループで CPU 時間を大幅に消費していることが分かります。外側のループの列をダブルクリックして、[Survey Source (調査ソース)] ビューの詳細を確認します。

シリアル・アプリケーションのモデリング



- ① この 2 つの入れ子されたループがプログラム実行時間の大きな割合を占めています。これらのループのデータアクセスと消費時間を評価すると、外側のループ本体が検証対象の有力候補であることが分かります。図の水平方向のラインを並列でレンダリングできます(図 8)。

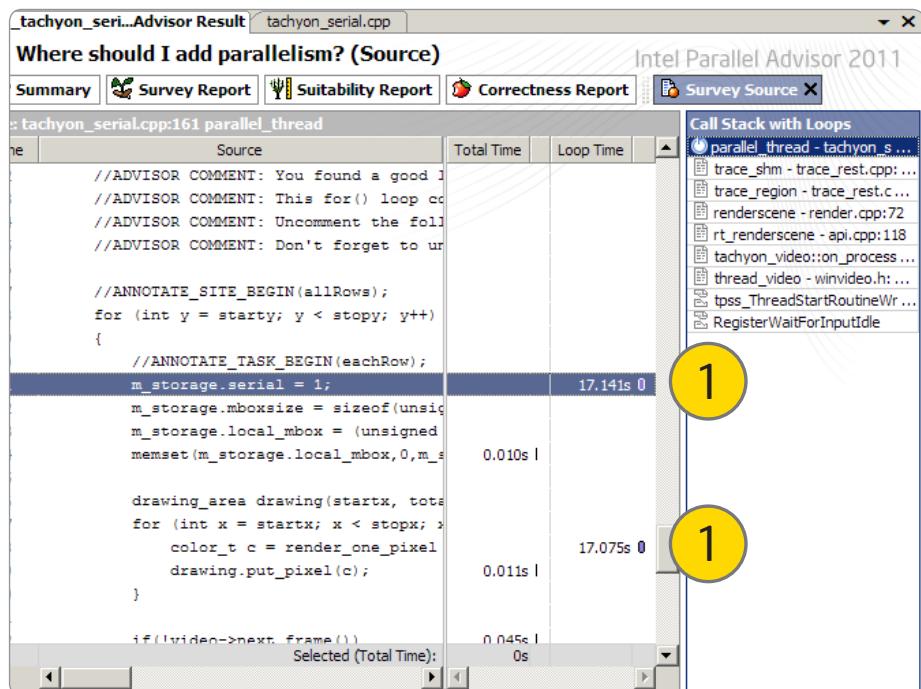
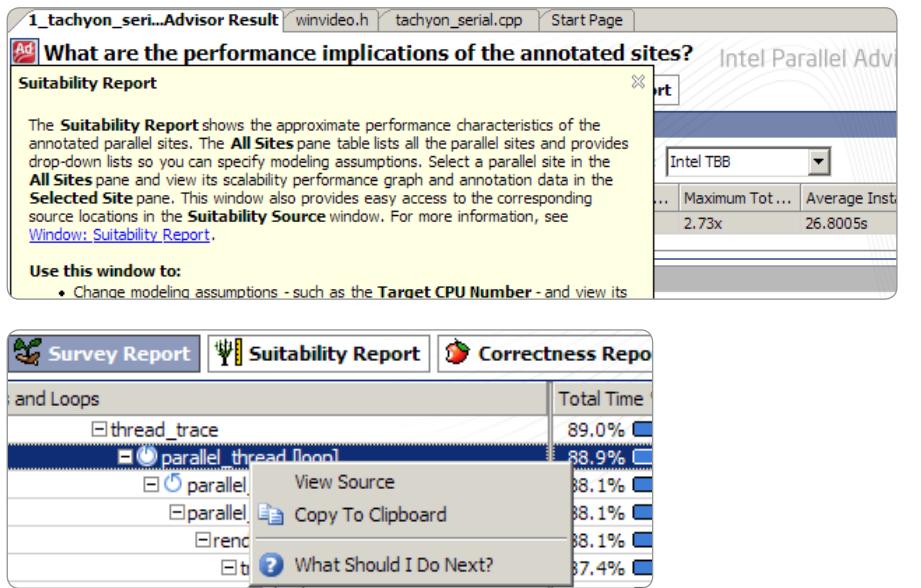


図 8

その他のアシスタンス

- ワークフロー・ウィンドウ上部にヘルプ ヒントが表示されます。
- ワークフローから、参考資料を確認できます。
- [My Advisor Results (マイ・アドバイザー 結果)] ウィンドウ上部の [Intel Parallel Advisor (インテル (R) Parallel Advisor)] アイコンをダブルクリックすると、概要が表示されます。
- [My Advisor Results (マイ・アドバイザー 結果)] ウィンドウの行または列を右クリックして、コンテキスト・メニューから [What should I do next? (次の操作)] を選択できます。



シリアル・アプリケーションのモデリング



ソースのアノテーション：インテル® Parallel Advisor のアノテーションを挿入して並列化をモデリング

並列タスクを作成すべき場所（外側のループ本体）が分かったので、インテル® Parallel Advisor を使用して、並列化した場合のアプリケーションのパフォーマンスをモデリングすることができます。このとき、アプリケーションはシリアルのまま検証できます。

インテル® Parallel Advisor で並列化の検証を定義するため、次に [2. Annotate Sources (2. ソースのアノテーション)] を行います。

[Explain (説明)] ボタンをクリックして、ソースのアノテーションに必要なステップがまとめられたダイアログボックスを開きます（図 9）。

次のいずれかの操作を行って、編集可能なコードウィンドウに移動します。

- › インテル® Parallel Advisor ウィンドウの行をダブルクリックします。
- › 右クリックして、コンテキストメニューから [Edit Source (ソースの編集)] を選択します。

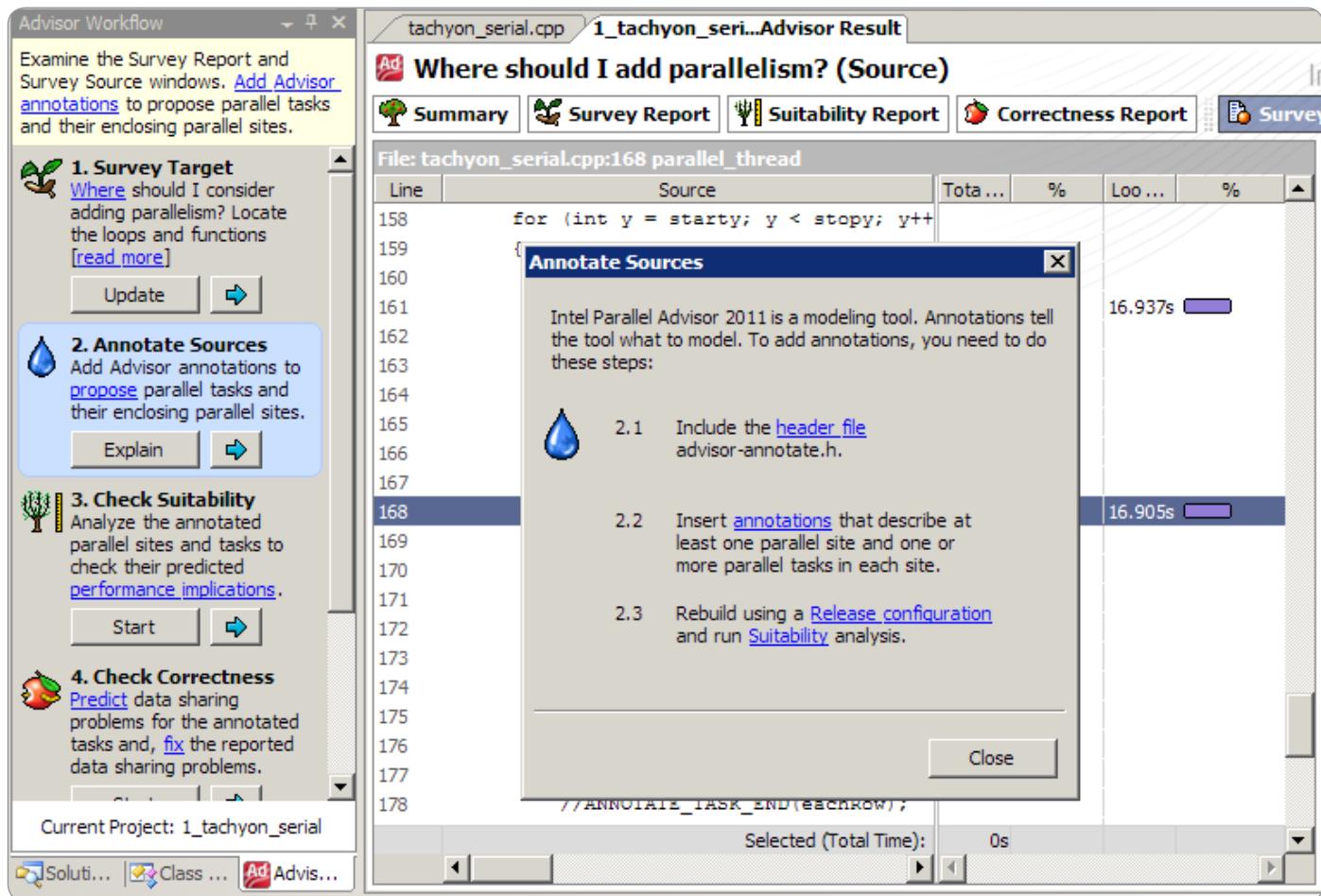
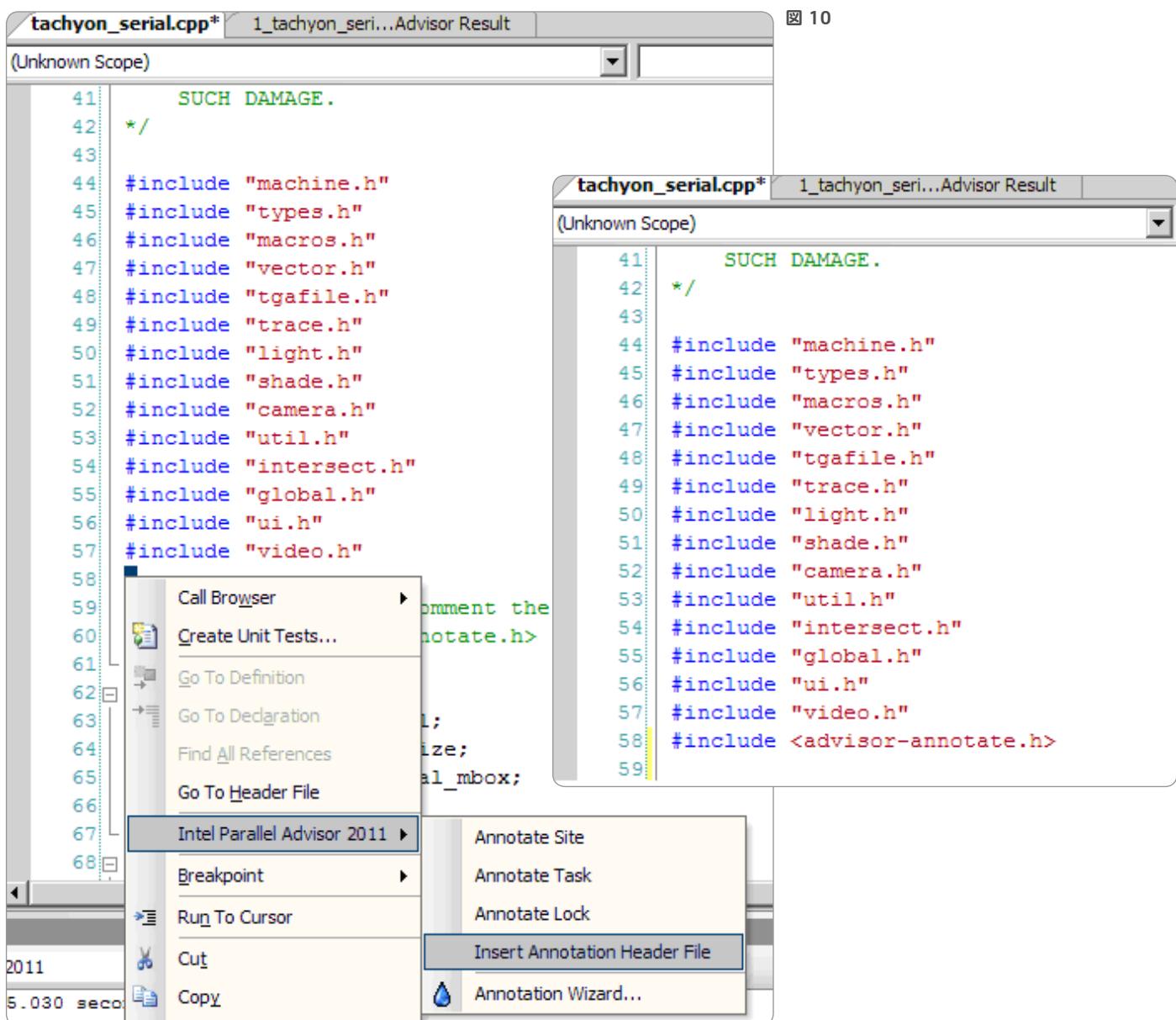


図 9

1. advisor-annotate.h ヘッダーファイルをインクルードします。

このヘッダーファイルは、インテル® Parallel Advisor の include ディレクトリーにあります。サンプル・プロジェクトのインクルード・パス・プロパティーはすでに変更されているため、ここではソースファイルにインクルード文を追加するだけです。

エディターウィンドウでヘッダーファイルをインクルードする場所に移動し、右クリックしてエディターのコンテキスト・メニューから [Intel Parallel Advisor 2011 (インテル (R) Parallel Advisor 2011)] > [Insert Annotation Header File (アノテーション・ヘッダー・ファイルの挿入)] を選択します (図 10)。



シリアル・アプリケーションのモデリング



2. 次に、タスク・アノテーションを挿入して並列タスクを定義します。エディターウィンドウで外側のループ `for (int y = starty; y < stopy; y++)` の本体を選択し、右クリックしてエディターのコンテキストメニューから [Intel Parallel Advisor 2011 (インテル (R) Parallel Advisor 2011)] > [Annotation Wizard (アノテーション・ウィザード)] を選択します (図 11)。

[Annotation Wizard (アノテーション・ウィザード)] を使用して、「MyTask1」というラベルのタスク・アノテーションを挿入します (図 12)。

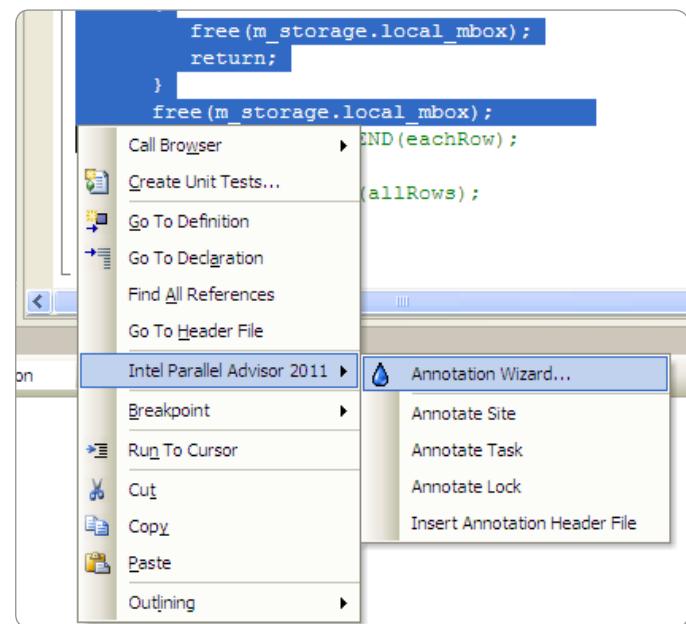
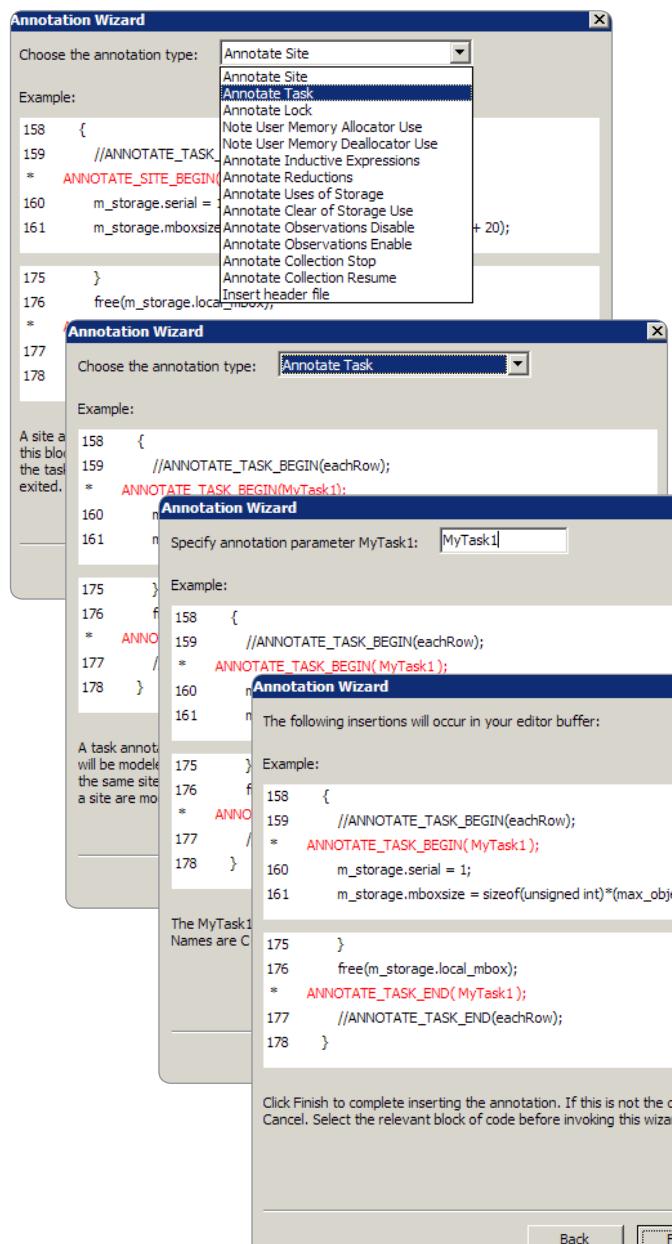


図 11

図 12

シリアル・アプリケーションのモデリング



3. 次に、タスクを並列領域のアノテーションで囲みます。再度、ループ全体を選択し、右クリックして、[Intel Parallel Advisor 2011 (インテル (R) Parallel Advisor 2011)] > [Annotate Site (領域のアノテーション)] を選択します。このアノテーションには、インテル® Parallel Advisor で生成されたラベルが使用されます (図 13)。

図 13

図 14

```

tachyon_serial.cpp* 1_tachyon_ser...Advisor Result
(Unknown Scope)
157 //ANNOATATE_SITE_BEGIN(allRows);
158 for (int v = starty; y < stopy; y++)
159     Call Browser
160     Create Unit Tests...
161     Go To Definition
162     Go To Declaration
163     Find All References
164     Go To Header File
165     Intel Parallel Advisor 2011 > Annotation Wizard...
166     Breakpoint
167     Run To Cursor
168     Cut
169     Copy
170     Paste
171     Outlining
172
173     //ANNOATATE_TASK_BEGIN(eachRow);
174     ANNOATATE_TASK_BEGIN( MyTask1 );
175     //ANNOATATE_TASK_END( eachRow );
176     //ANNOATATE_SITE_END(allRows);
177 }
178 free(m_storage.local_mbox);
179 ANNOATATE_TASK_END( MyTask1 );
180 //ANNOATATE_TASK_END( eachRow );
181
182 //ANNOATATE_SITE_END(allRows);

1
1
1
1

```

1 插入されたインテル® Parallel Advisor のアノテーション (図 14)。

4. プロジェクトをビルドします。

シリアル・アプリケーションのモデリング



適合性の確認：パフォーマンスへの影響は？

インテル® Parallel Advisor の適合性検証ツールは、並列領域のパフォーマンス予測とプログラム全体への影響を表示することで、並列化により得られるパフォーマンスの評価を支援します。適合性検証ツールでプログラムを解析するには、[3. Check suitability (3. 適合性の確認)] にある [Start (開始)] を選択します（図 15）。

1_tachyon_serial...Advisor Result tachyon_serial.cpp

Ad What are the performance implications of the annotated sites?

Summary Survey Report Suitability Report Correctness Report

Slowdown

The Suitability tool analyzes your running program. To predict the serial program's likely parallel behavior, it carefully examines the proposed parallel sites and tasks that you identified with Advisor annotations.

並列化によってメリットが得られそうですね。次に、データ共有の問題を確認する必要があります。

1_tachyon_serial...Advisor Result tachyon_serial.cpp

Ad What are the performance implications of the annotated sites?

Summary Survey Report Suitability Report Correctness Report

All Sites

2 Maximum Program Gain For All Sites: 2.83x

Annotation ...	Source Location	Maximum Sit...	Maximum Tot...	Average Insta...	Total T...
MySite1	tachyon_serial...	7.75x	2.83x	28.9885s	28.98...

3 Scalability of Maximum Site Gain

4

Annotation ...	Source Location	Number of Ins ...	Maximum Insta...	Average Instan...	Minimum Instan...	Devia...	Total T...
MySite1	tachyon_serial...	1	28.9885s	28.9885s	28.9885s	0%	28.9885s
Task MyTask1	tachyon_serial...	512	0.0560s	0.0560s	0.0560s	0%	28.6497s

5 Changes I will make to this site to improve performance

Type of Change	Benefit	Recommendation
<input type="checkbox"/> Reduce Site Overhead	0x	How Do I Reduce Site Overhead?
<input type="checkbox"/> Reduce Task Overhead	0x	How Do I Reduce Task Overhead?
<input type="checkbox"/> Reduce Lock Overhead	0x	How Do I Reduce Lock Overhead?
<input type="checkbox"/> Reduce Lock Contention	0x	How Do I Reduce Lock Contention?
<input type="checkbox"/> Enable Task Chunking	0x	How Do I Enable Task Chunking?

1

並列領域

4

選択された領域内で実行されたアノテーション

2

プログラム全体への影響

5

選択された領域のパフォーマンスを最大限に引き出す方法

3

選択された領域のスケーラビリティー

シリアル・アプリケーションのモデリング



正当性の確認：潜在的なデータ共有問題があるかどうか

インテル® Parallel Advisor の正当性検証ツールは、並列領域のデータの問題（競合）を特定するのに役立ちます。Debug 構成に切り替えてビルドしてから、[4. Check Correctness (正当性の確認)] にある [Start (開始)] を選択します（図 16）。

正当性検証ツールを実行すると、このサンプル・アプリケーションには、並列化する場合に解決しなければならないデータ共有問題があることが分かります（図 17）。

図 16

正当性の予測

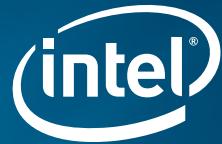
正当性検証ツールは、データ共有や関連する問題を予測するために実行中のシリアルプログラムの広範な解析を行います。問題が検出された場合は、次のいずれかの操作を行って、編集可能なコードウィンドウに移動します。

- インテル® Parallel Advisor ウィンドウの行をダブルクリックします。
- 右クリックして、コンテキスト・メニューから [Edit Source (ソースの編集)] を選択します。

- ① データ収集の進捗状況
- ② 検出された問題とメッセージ
- ③ 検出された問題とメッセージの詳細

図 17

シリアル・アプリケーションのモデリング



全体像の確認

いくつかの並列モデル（複数の領域やタスク）を検証する際には、サマリーレポートを参照すると良いでしょう。このレポートには、スタートアップ・プロジェクトが実行されたときに発生したすべてのアノテーションがリストされ、それぞれのパフォーマンス予測とデータ共有問題の概要が表示されます。

インテル® Parallel Advisor ウィンドウの [Summary (概要)] を選択すると、次のような画面が表示されます。

The screenshot shows the Intel Parallel Advisor 2011 interface with the title bar "tachyon_serial.cpp 1_tachyon_ser...Advisor Result". Below it is the sub-title "Summary of predicted parallel behavior". There are four tabs: "Summary" (selected), "Survey Report", "Suitability Report", and "Correctness Report". The main content area displays a table with the following data:

Annotation	Source Location	Annotation Label	Self Time	Maximum Self Gain	Maximum Total Gain	Correctness Problems
+ Site	tachyon_serial.cpp:1...	MySite1	28.9885s	7.75x	2.83x	4 errors 0 warnings
+ Site End	tachyon_serial.cpp:1...	MySite1	-	-	-	--
+ Task	tachyon_serial.cpp:1...	MyTask1	28.6497s	-	-	--
+ Task End	tachyon_serial.cpp:1...	MyTask1	-	-	-	--
+ Annotation	tachyon_serial.cpp:58	advisor-annotate.h	-	-	-	--

At the bottom, there is a note: "Modeling Assumptions: Target CPU Number: 8; Threading Model: Intel TBB".

それぞれの問題行をダブルクリックして、対応する [Correctness Source (正当性ソース)] ビューを表示し、データ共有問題を調査します。最初の行 (P1) の「memory reuse (メモリーの再利用)」に関する問題から見てみましょう（図 18）。

この問題と別のメモリーの再利用の問題 (P2 と P3) と比較すると、この 3 つの問題は関連しており、`m_storage` の偶発的な共有によるものであることが分かります。並列バージョンが正しく動作するには、それぞれのタスク（ループの反復）ごとに `m_storage` のコピーを持つ必要があります。これは、`m_storage` の宣言をタスク（ループ）内に移動することで可能です。

The screenshot shows the Intel Parallel Advisor 2011 interface with the title bar "1_tachyon_ser...Advisor Result" and sub-title "Did the annotated tasks expose data sharing problems? (Source)". Below it is the tab "Correctness Source" (selected). The main content area displays two code snippets and a call stack:

- Focus Observation: tachyon_serial.cpp:164 - Write**

```
162     // ANNOTATE_TASK_BEGIN(eachRow);
163     ANNOTATE_TASK_BEGIN( MyTask1 );
164     m_storage.serial = 1;
165     m_storage.mboxsize = sizeof(unsigned int)*(max_object
166     m_storage.local_mbox = (unsigned int *) malloc(m_stor
167     memset(m_storage.local_mbox, 0, m_storage.mboxsize);
168 
```
- Related Observation: tachyon_serial.cpp:97 - Write**

```
95         primary.flags = RT_RAY_REGULAR;
96
97         serial++;
98         primary.serial = serial;
99         primary.mbox = local_mbox;
100        primary.maxdist = FHUGE;
101        primary.scene = scene;
```
- Call Stack**
 - parallel_thread - tachyon_serial.cpp:164
 - render_one_pixel - tachyon_serial.cpp:97
 - render_one_pixel - tachyon_serial.cpp:84
 - parallel_thread - tachyon_serial.cpp:171
 - thread_trace - tachyon_serial.cpp:201
 - trace_shm - trace_rest.cpp:104
 - trace_region - trace_rest.cpp:117
 - renderscene - render.cpp:87

Below these are two tables:

- Memory reuse: Observations**

ID	Description	Source	Function	Module	State
X4	Parallel site	tachyon_ser...	parallel_thread	1_tachyon_ser...	Info
X5	Write	tachyon_ser...	render_one_...	1_tachyon_ser...	Not fi...
X6	Write	tachyon_ser...	parallel_thread	1_tachyon_ser...	Not fi...
X7	Read	tachyon_ser...	render_one_...	1_tachyon_ser...	Not fi...
- Relationship Diagram**

A diagram showing a relationship between two writes: "Write tachyon_serial.cpp:97" and "Write tachyon_serial.cpp:164".

図 18

シリアル・アプリケーションのモデリング



1_tachyon_serial...Advisor Result tachyon_serial.cpp Start Page

Did the annotated tasks expose data sharing problems? (Source)

Summary Survey Report Suitability Report Correctness Report Correctness Source X

Focus Observations: tachyon_serial.cpp:164 - Write

```

162     //ANNOTATE_TASK_BEGIN(eachRow);
163     ANNOTATE_TASK_BEGIN( MyTask1 );
164     m_storage.serial = 1;
165     m_storage.mboxsize = sizeof(unsigned int)*(max_objectid() + 20);
166     m_storage.local_mbox = (unsigned int *) malloc(m_storage.mboxsize);
167     memset(m_storage.local_mbox,0,m_storage.mboxsize);
168

```

Related Observation: tachyon_serial.cpp:97 - Write

```

95     primary.flags = RT_RAY_REGULAR;
96
97     serial++;
98     primary.serial = serial;
99     primary.mbox = local_mbox;
100    primary.maxdist = FHUGE;
101    primary.scene = scene;

```

Memory reuse: Observations

ID	Description	Source	Function	Module	State
X4	Parallel site	tachyon_serial...	parallel_thread	1_tachyon_serial...	Informational
X5	Write	tachyon_serial...	render_one_...	1_tachyon_serial...	Not found
X6	Write	tachyon_serial...	parallel_thread	1_tachyon_serial...	Not found
X7	Read	tachyon_serial...	render_one_...	1_tachyon_serial...	Not found

Call Stack

```

parallel_thread - tachyon_serial.cpp:164
render_one_pixel - tachyon_serial.cpp:97
render_one_pixel - tachyon_serial.cpp:84
parallel_thread - tachyon_serial.cpp:171
thread_trace - tachyon_serial.cpp:201
trace_shm - trace_rest.cpp:104
trace_region - trace_rest.cpp:117
renderscene - render.cpp:87

```

Relationship Diagram

```

graph LR
    A[Write tachyon_serial.cpp:97] --> B[Write tachyon_serial.cpp:164]

```

図 19

[Observation (問題箇所)] ウィンドウの行を右クリックして、[Edit Source (ソースの編集)] を選択し、エディターウィンドウに移動して、`m_storage` の宣言 (図 20) をタスク内部に移動します (図 21)。

1_tachyon_serial...Advisor Result tachyon_serial.cpp Start Page

(Global Scope)

```

151 static void parallel_thread (void)
152 {
153     //ADVISOR COMMENT: You found a good location to model parallelism
154     //ADVISOR COMMENT: This for() loop consumes a lot of the runtime and is a prime
155     //ADVISOR COMMENT: Uncomment the following four annotations to model the iteration
156     //ADVISOR COMMENT: Don't forget to uncomment the #include <advisor-annotate.h> etc.
157
158     //ANNOTATE_SITE_BEGIN(allRows);
159     ANNOTATE_SITE_BEGIN(MySite3);
160     for (int y = starty; y < stopy; y++)
161     {
162         //ANNOTATE_TASK_BEGIN(eachRow);
163         ANNOTATE_TASK_BEGIN( MyTask1 );
164         storage m_storage;
165         m_storage.serial = 1;
166         m_storage.mboxsize = sizeof(unsigned int)*(max_objectid() + 20);
167         m_storage.local_mbox = (unsigned int *) malloc(m_storage.mboxsize);
168         memset(m_storage.local_mbox,0,m_storage.mboxsize);
169
170         drawing_area drawing(startx, totaly-y, stopx-startx, 1);
171         for (int x = startx; x < stopx; x++) {
172             color_t c = render_one_pixel (x, y, m_storage.local_mbox, m_storage.ser:

```

図 21

シリアル・アプリケーションのモデリング

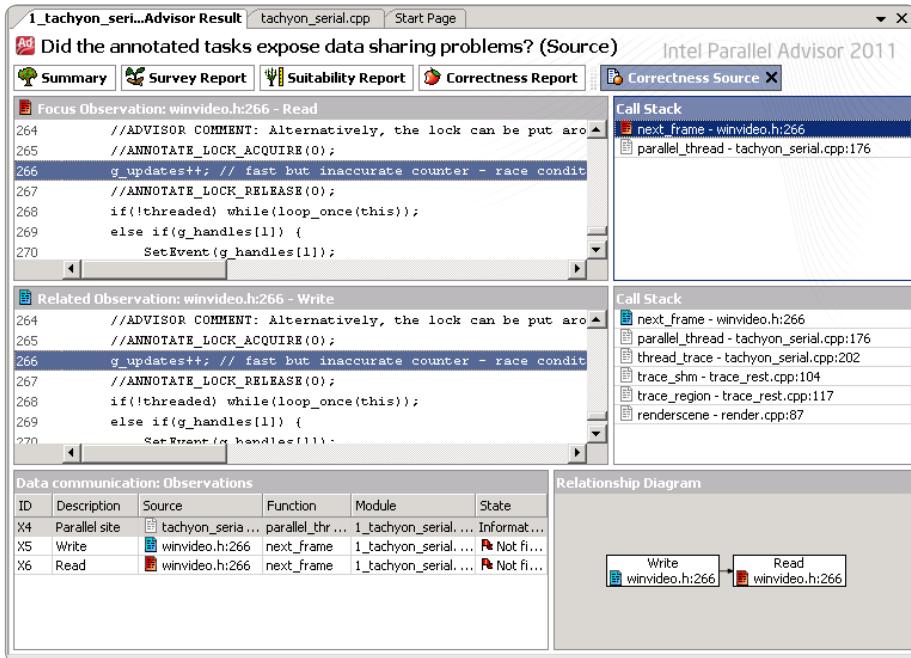


図 22

検出された最後の問題はデータ通信 (P4) です。調査の結果、ループの反復間で一貫して更新される必要のあるグローバル変数 `g_updates` があることがわかりました。この問題は、ローカルコピーの定義とインクリメントでは解決できません (図 22)。

簡単な解決方法は、`g_updates` の更新の周囲にロック・アノテーションのペアを挿入することです。編集可能なソースウィンドウに移動し、ソース行を選択して右クリックし、[Intel Parallel Advisor 2011 (インテル(R)) Parallel Advisor 2011] > [Annotate Lock (ロックのアノテーション)] を選択します (図 23)。

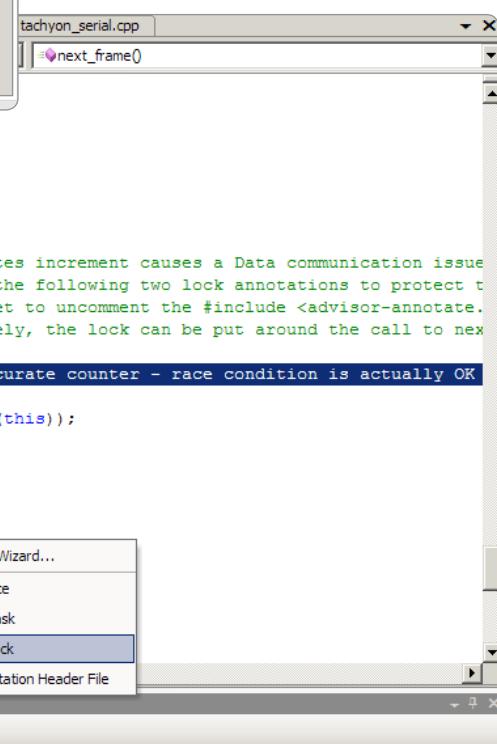


図 23

```

255 }
256 //! Refresh screen picture
257 bool video::next_frame()
258 {
259     if(!running) return false;
260     //ADVISOR COMMENT: The g_updates increment causes a Data communication issue
261     //ADVISOR COMMENT: Uncomment the following two lock annotations to protect t
262     //ADVISOR COMMENT: Don't forget to uncomment the #include <advisor-annotate.
263     //ADVISOR COMMENT: Alternatively, the lock can be put around the call to nex
264     //ANNOTATE_LOCK_ACQUIRE(0);
265     g_updates++; // fast but inaccurate counter - race condition is actually OK
266     Call Browser    ▶ RELEASE(0);
267     Create Unit Tests... ▶ e(loop_once(this));
268     Go To Definition ▶ 1];
269     Go To Declaration ▶ files[1];
270     Find All References
271     Intel Parallel Advisor 2011 ▶
272         Annotation Wizard...
273         Annotate Site
274         Annotate Task
275         Annotate Lock
276         Insert Annotation Header File
277     Breakpoint ▶
278     Run To Cursor
279     Cut
280     Copy
281     Paste
282     Outlining ▶
283
284     257 //! Refresh screen picture
285     258 bool video::next_frame()
286     259 {
287     260     if(!running) return false;
288     //ADVISOR COMMENT: The g_updates increment causes a Data communication issue
289     //ADVISOR COMMENT: Uncomment the following two lock annotations to protect t
290     //ADVISOR COMMENT: Don't forget to uncomment the #include <advisor-annotate.
291     //ADVISOR COMMENT: Alternatively, the lock can be put around the call to nex
292     //ANNOTATE_LOCK_ACQUIRE(0);
293     ANNOTATE_LOCK_ACQUIRE(0);
294     g_updates++; // fast but inaccurate counter - race condition is actually OK
295     ANNOTATE_LOCK_RELEASE(0);
296     //ANNOTATE_LOCK_RELEASE(0);
297     if(!threaded) while(loop_once(this));
298     else if(g_handles[1]) {
299         SetEvent(g_handles[1]);
300         YIELD_TO_THREAD();
301     }
302     return true;
303 }

```

シリアル・アプリケーションのモデリング



ソースの変更による影響の確認

正当性のデータ問題を解決したら、ビルドし、適合性検証ツールと正当性検証ツールを再度実行して、ソースの変更による影響を評価し、新しい問題が発生していないか確かめます。結果は次のようにになります。

- > 正当性検証ツールでは、新たな問題は特定されないでしょう。
- > 適合性検証ツールでは、g_updates のロックは最小限の影響のみにとどまるこことを再確認できるでしょう（図 24）。

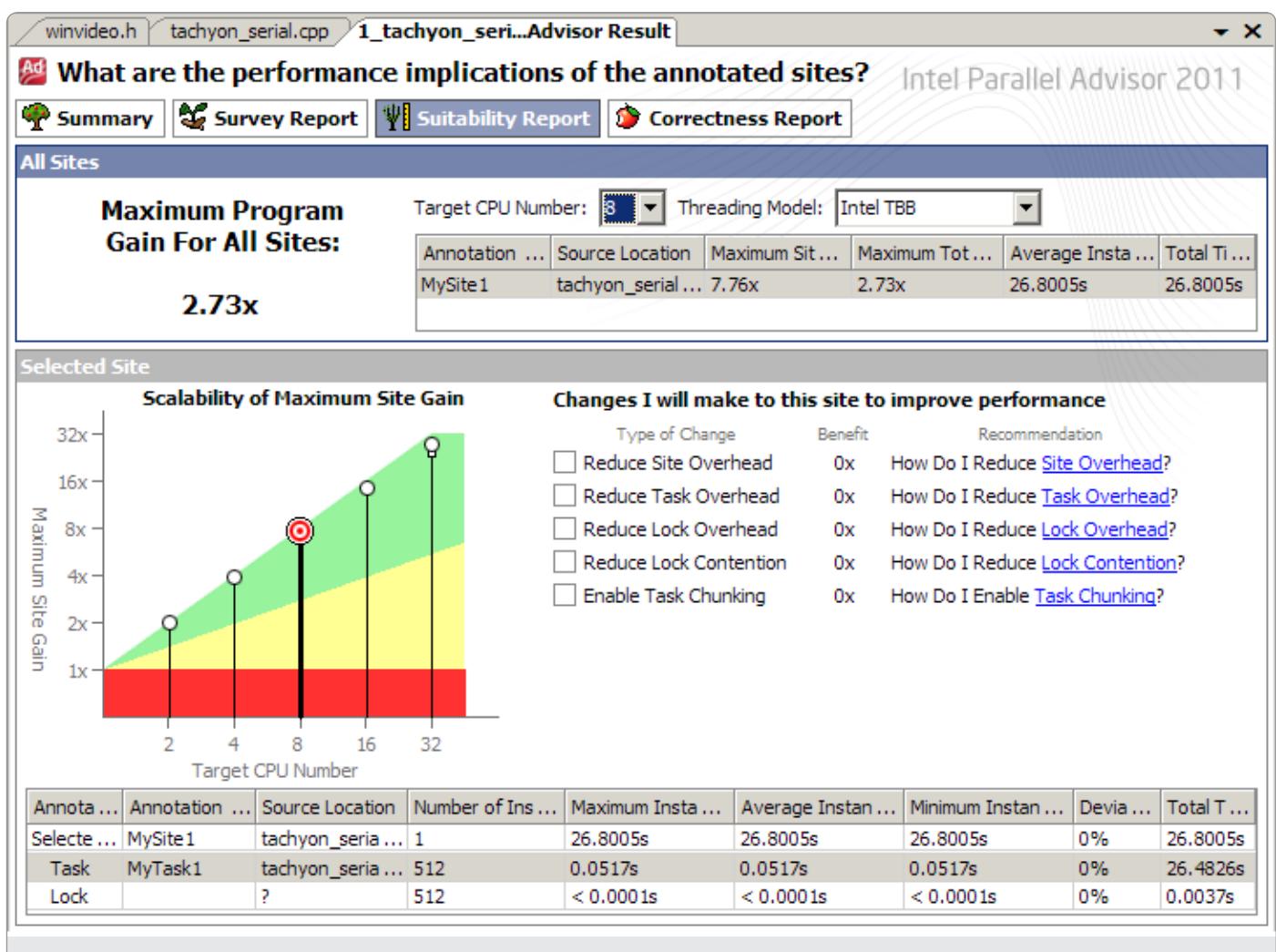


図 24

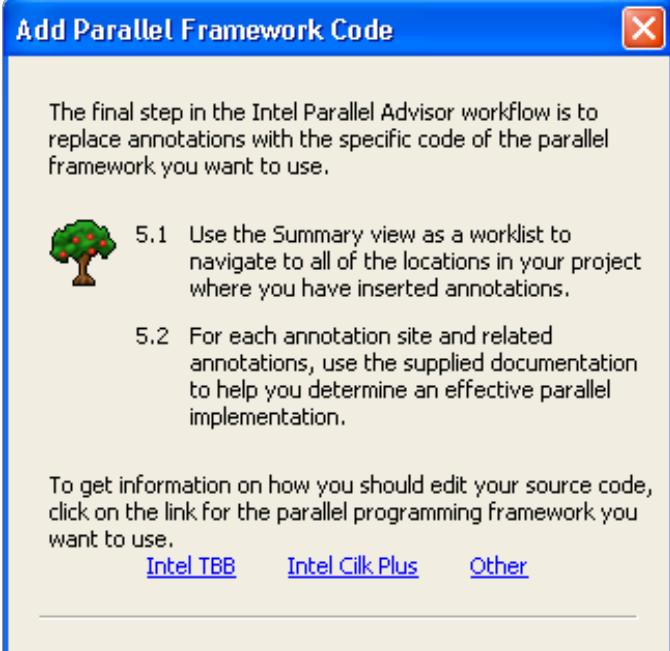
シリアル・アプリケーションのモデリング



アノテーションから並列フレームワークへの移行

最後に、インテル® Parallel Advisor のアノテーションを並列フレームワークに移行します。これは、インテル® Parallel Advisor の [5. Add Parallel Framework (5. 並列フレームワークの追加)] にあたります。ウィンドウにある [Explain (説明)] ボタンをクリックします (図 25)。

インテル® Parallel Advisor には、アノテーションから並列フレームワークへの移行に役立つドキュメントが用意されています。サマリービューの行をダブルクリックするか、右クリックして [Edit Source (ソースの編集)] を選択すると、それぞれのアノテーションに素早く移動できます (図 26)。



winvideo.h tachyon_serial.cpp 1_tachyon_ser... Advisor Result

Summary Survey Report Suitability Report Correctness Report

Intel Parallel Advisor 2011

Close

Summary of predicted parallel behavior

Annotation	Source Location	Annotation Label	Self Time	Maximum Self Gain	Maximum Total Gain	Correctness Problems
+ Site	tachyon_serial.cpp: ...	MySite1	26.8005s	7.76x	2.73x	0 errors 0 warnings
+ Site End	tachyon_serial.cpp: ...	MySite1	-	-	-	- -
Task	tachyon_serial.cpp: ...	MyTask1	26.4826s	-	-	- -
<pre> 161 { 162 //ANNOVATE_TASK_BEGIN(eachRow); 163 ANNOVATE_TASK_BEGIN(MyTask1); 164 storage m_storage; 165 m_storage.serial = 1; </pre>						
+ Task End	tachyon_serial.cpp: ...	MyTask1	-	-	-	- -
Unmatched L ... ?	-	-	0.0103s	-	-	- -
+ Lock	winvideo.h:266	0	?	-	-	- -
+ Lock Release	winvideo.h:268	0	-	-	-	- -
+ Annotation	winvideo.h:37	advisor-annotate.h	-	-	-	- -
+ Annotation	tachyon_serial.cpp:58	advisor-annotate.h	-	-	-	- -

Modeling Assumptions: Target CPU Number: 8; Threading Model: Intel TBB

図 26

並列バージョンのビルドと速度向上の確認

インテル® Parallel Advisor アノテーションをインテル® スレッディング・ビルディング・ブロック (インテル® TBB) などの適切な並列フレームワークに置き換えたら、マルチコアシステムでビルドし、実行します。3_tachyon_cilk プロジェクトまたは 3_tachyon_tbb プロジェクトで、インテル® TBB またはインテル® Cilk™ Plus を使用して tachyon を並列化した結果を確認できます。

以下の画像は、4 コアのシステムで tachyon を実行した場合を示しています。最初、レンダリングは、4 つに分かれて表示されますが、その後、「thread stealing (スレッドのスチール)」により完了します (図 27)。

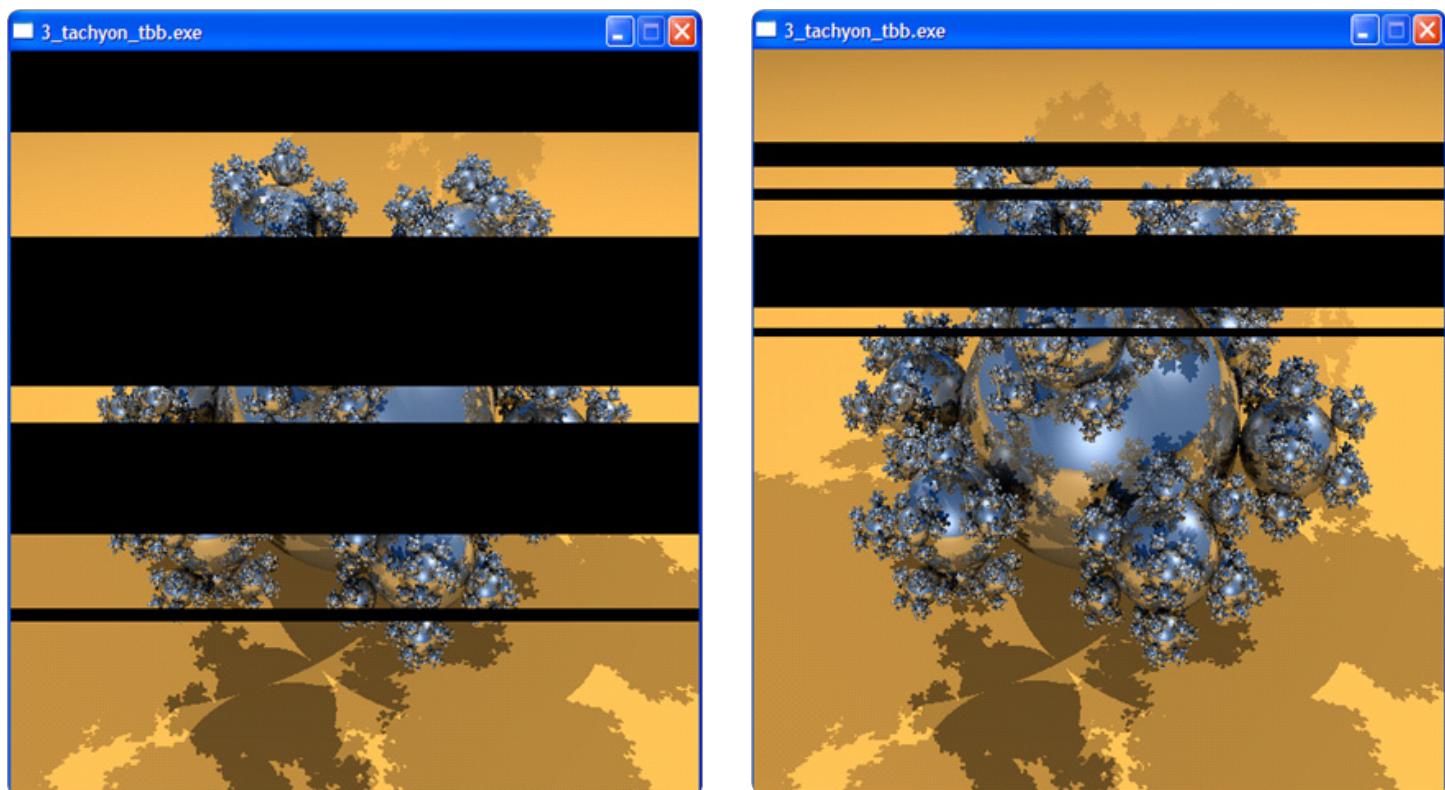


図 27

結果

この例は、インテル® Parallel Advisor を使用して、既存のアプリケーションの並列化を設計する方法を示しています。インテル® Parallel Advisor は、変更が容易なシリアルコードにおいて、潜在的なデータ競合やその他の問題を特定しながら、並列化によるパフォーマンスの恩恵を評価するのに役立ちます。

シリアル・アプリケーションのモデリング



最適化に関する注意事項

インテル® コンパイラ、関連ライブラリーおよび関連開発ツールには、インテル製マイクロプロセッサーおよび互換マイクロプロセッサーで利用可能な命令セット (SIMD 命令セットなど) 向けの最適化オプションが含まれているか、あるいはオプションを利用している可能性がありますが、両者では結果が異なります。また、インテル® コンパイラ用の特定のコンパイラ・オプション (インテル® マイクロアーキテクチャーに非固有のオプションを含む) は、インテル製マイクロプロセッサー向けに予約されています。これらのコンパイラ・オプションと関連する命令セットおよび特定のマイクロプロセッサーの詳細は、『インテル® コンパイラ・ユーザー・リファレンス・ガイド』の「コンパイラ・オプション」を参照してください。インテル® コンパイラ製品のライブラリー・ルーチンの多くは、互換マイクロプロセッサーよりもインテル製マイクロプロセッサーでより高度に最適化されます。インテル® コンパイラのコンパイラとライブラリーは、選択されたオプション、コード、およびその他の要因に基づいてインテル製マイクロプロセッサーおよび互換マイクロプロセッサー向けに最適化されますが、インテル製マイクロプロセッサーにおいてより優れたパフォーマンスが得られる傾向にあります。

インテル® コンパイラ、関連ライブラリーおよび関連開発ツールは、互換マイクロプロセッサー向けには、インテル製マイクロプロセッサー向けと同等レベルの最適化を行わない可能性があります。これには、インテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2)、インテル® ストリーミング SIMD 拡張命令 3 (インテル® SSE3)、ストリーミング SIMD 拡張命令 3 補足命令 (インテル® SSSE3) 命令セットに関連する最適化およびその他の最適化が含まれます。インテルでは、インテル製ではないマイクロプロセッサーに対して、最適化の提供、機能、効果を保証していません。本製品のマイクロプロセッサー固有の最適化は、インテル製マイクロプロセッサーでの使用を目的としています。

インテルでは、インテル® コンパイラおよびライブラリーがインテル製マイクロプロセッサーおよび互換マイクロプロセッサーにおいて、優れたパフォーマンスを引き出すのに役立つ選択肢であると信じておりますが、お客様の要件に最適なコンパイラを選択いただくよう、他のコンパイラの評価を行うことを推奨しています。インテルでは、あらゆるコンパイラーやライブラリーで優れたパフォーマンスが引き出され、お客様のビジネスの成功のお役に立ちたいと願っております。お気づきの点がございましたら、お知らせください。

改訂 #20101101



並列処理に関する情報

インテルでは、開発者が現在および将来のプロセッサー処理能力を活用する、正当で高性能なコードを記述できるように、並列処理に関するさまざまな情報を提供しています。並列化やインテル® Parallel Studioについての詳細は、以下を参照してください。

関連リンク (英語)

- [インテル® ソフトウェア・ネットワーク・フォーラム](#) 
- [インテル® ソフトウェア開発製品ナレッジベース](#) 
- [インテル® ソフトウェア・ネットワーク・ブログ](#) 
- [インテル® Parallel Studio Web サイト](#) 
- [インテル® TBB Web サイト](#) 
- [並列化に関するブログ、記事、ビデオ](#) 
- [開発者向け Web セミナー \(オンデマンド\)](#) 